An incentive compatible reputation mechanism for ubiquitous computing environments

Jinshan Liu, Valérie Issarny

An Incentive Compatible Reputation Mechanism for Ubiquitous Computing Environments

Jinshan Liu
INRIA - Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105
78153 Le Chesnay Cedex, France
jinshan.liu@inria.fr

Valérie Issarny
INRIA - Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105
78153 Le Chesnay Cedex, France
valerie.issarny@inria.fr

ABSTRACT

The vision of *ubiquitous computing* is becoming a reality thanks to the advent of portable devices and the advances in wireless networking technologies. It aims to facilitate user tasks through seamless utilization of services available in the surrounding environments. In such distributed environments featuring openness, interactions, especially service provision and consumption, between entities that are unknown or barely known to each other, are commonplace. Trust management through reputation mechanism to facilitate such interactions is recognized as an important element of ubiquitous computing. It is, however, faced by the problems of how to stimulate reputation information sharing and honest recommendation elicitation. We present in this paper an incentive compatible reputation mechanism to facilitate the trustworthiness evaluation in ubiquitous computing environments. It is based on probability theory and supports reputation evolution and propagation. Our reputation mechanism not only shows robustness against lies, but also stimulates honest and active recommendations. The latter is realized by ensuring that active and honest recommenders, compared to inactive or dishonest ones, can elicit the most honest (helpful) recommendations and thus suffer the least number of wrong trust decisions, as validated by simulation based evaluation.

Categories and Subject Descriptors

D.2.0 [**SOFTWARE ENGINEERING**]: General—*Protection mechanisms*; C.2.4 [**COMPUTER COMMUNICATION NETWORKS**]: Distributed Systems—*Distributed applications*

General Terms

Security

Keywords

Reputation mechanism, ubiquitous computing, incentive compatibility, mobile ad hoc networks

PST 2006 Markham, Canada

1. INTRODUCTION

With the advent of portable devices (e.g., smartphones) and the advances in wireless networking technologies (e.g., WLAN, GPRS, UMTS), the vision of *ubiquitous computing* [29] is becoming a reality. It refers to the creation of environments saturated with a spectrum of heterogeneous computing and communication capabilities, which seamlessly integrate with the physical world [27]. It aims to facilitate daily tasks and enhance user productivity through the utilization of those capabilities in an unobtrusive fashion, such that they completely blend in the physical environment and become "invisible".

The heterogeneous capabilities available in the ubiquitous computing environment can be generalized as *services*, leading to Service oriented Computing (SoC), which is a computing paradigm that utilizes services as fundamental elements for developing applications [23]. A device carried by a user can be a *service client*, which is an entity in need of services; or a *service provider*, which is an entity that offers services. SoC fits ubiquitous computing thanks to its *minimalist philosophy* [28], i.e., an entity only needs to carry a small amount of codes locally and discover and exploit other services to realize its tasks. A service, a set of functionalities provided by one entity for the use of others [22], is characterized by its functional and non-functional attributes.

The networking between devices can be through network infrastructure (e.g., a home wireless LAN), which is assumed to be always accessible for nomadic mobile devices. However, it requires deployment and maintenance and cannot assume to be always available. Therefore, in order to achieve "all the time everywhere" access to services in ubiquitous computing environments, it necessitates a more flexible alternative for networking. Mobile Ad hoc NETworks (MANET) pose as a good choice: mobile devices dynamically establish connections with others when needed [17]. The devices (nodes) are free to move around and the network can be reorganized arbitrarily. In contrast with infrastructured networks, MANETs are deployment-free and realize spontaneous networking of devices. Thus they support impromptu interaction between entities, which is a desirable feature for ubiquitous computing [12]. Hence, with MANETs, a user equipped with her device, even when she is moving, can dynamically find and exploit the services available in the surrounding environments, which are not necessarily pre-deployed. In summary, MANET poses as a flexible and suitable underlying networking paradigm for ubiquitous

computing environments. Therefore, in this paper, entities are assumed to be connected via ad hoc networks. The terms of *device*, *node* and *entity* are used interchangeably in the rest of this paper.

Overall, the service provision in ubiquitous computing environments has the following characteristics:

- There does not exist any party that is centralized or pre-trusted, due to the independence of any infrastructure to allow for more flexible interactions.
- The devices exhibit great mobility. It makes nodes' joining and leaving of a network much more frequent than in traditional wired networks, thus increasing the network's openness. Subsequently, it is very likely for an entity to encounter others, which it has no or very little knowledge of.
- Devices can be selfish because providing services consumes limited resources. Therefore, services bear prices, which are charged by service providers on clients, in order to enforce cooperation between devices. Different services have different prices depending on factors such as the offered QoS.

Before interacting with a service provider, a client needs to evaluate its trustworthiness, because a dishonest service provider can cheat (e.g., exaggerate its offered QoS) for more revenues. Traditional security mechanisms such as authentication and access control (e.g., X.509 [2]) fall short for the above purpose because of their reliance on security infrastructure such as Certificate Authority. More importantly, as it is commonplace to interact with strangers in ubiquitous computing environments, even with authentication and authorization services at disposal, authenticating an unknown entity does not provide any access control information. Trust and reputation, on the other hand, can provide protection against such threats. *Trust* deals with the estimation of a node's future behavior. For example, a client trusts a service provider in that the latter *will* actually offer the QoS as claimed. Trust is generally difficult to establish between strangers [25], because they do not have any previous experiences and are not subject to a network of informed entities about their behaviors. Reputation, which is "perception that a node creates through past actions about its intentions and norms" [21], is important for fostering trust [25], because it dissuades entities to misbehave resulting from no fear for future revenge. It has been proved to be a useful model and widely deployed in various scenarios such as electronic market places (e.g., eBay[1]). The reputation assessment of a trusted node, named *trustee*, by a trusting node, named *trustor* are dependent on [31]: (i) the trustor's own direct experiences with the trustee; (ii) the trustor's indirect experiences, i.e., *recommendations* (also named *ratings*) from other entities. The entities that give recommendations are called *recommenders*. To prevent loops, recommendations are only based on recommenders' own direct experiences.

Given the openness of the environments, it is very likely that before interacting with an entity, the accumulated direct experiences are too few or too old to derive a trust decision. Recommendations are thus indispensable for alleviating the above problem. However, recommendations can be difficult to elicit, i.e., entities are reluctant to recommend. This is because [20]: entities may be reluctant to give positive recommendations because they lift the reputation of the trustees, which are potential competitors; entities may be afraid of retaliation for negative feedbacks; last but not least, the (truthful) recommendations only benefit others. Meanwhile, recommendations are also subject to manipulation and can be false, e.g., colluders give high recommendations for each other. A false recommendation is called a *lie*. Since truthful recommendations are critical for a reputation mechanism to operate effectively [25], the two above issues pose obstacles for designing a reputation mechanism that is capable of recognizing the real trustworthiness of an entity.

Existing reputation mechanisms (e.g., [10, 30, 24]) do not solve the two aforementioned problems altogether. Therefore, we propose a distributed reputation mechanism that motivates entities to recommend truthfully and actively. Our mechanism empowers an entity to distinguish (1) between trustworthy and untrustworthy service providers and (2) between honest and dishonest recommenders. It not only shows robustness against lies, but also stimulates active and truthful recommendations. The latter is achieved by enforcing that honest and active recommenders can benefit more from others, while liars are identified and isolated.

In the rest of the paper, Section 2 surveys related work on distributed reputation systems. Section 3 shows the representation of reputation based on Beta distribution. Then we explain how the reputation is formed based on direct and indirect experiences (i.e., recommendations) in Section 4. It is followed by reputation evolution in Section 5. Then we proceed to present the propagation of reputation and the incentives for active and honest recommendation provision in Section 6. In Section 7, the proposed reputation mechanism is evaluated with respect to its different treatment for recommenders of different honesty and activeness. This paper finishes with concluding remarks.

2. RELATED WORK

Reputation mechanism has been widely used and deployed in online service provision (e.g., ebay), peer-to-peer systems (e.g., [11]) and mobile ad hoc networking (e.g., [19]). During online service provision, especially e-commerce, it is commonplace for parties that are unknown to each other to interact [25]. This opens up an issue of lack of trust between two parties before an interaction takes place. P2P networks (e.g., Gnutella[2]) are subject to attacks from anonymous malicious peers, such as virus spreading and fake file attack [11]. In mobile ad hoc networks, since nodes rely on the service of "packet forwarding" provided by their neighbors in order to communicate with others that are out of their communication range, reputation is necessary for evaluating a node's degree of being cooperative (i.e., in forwarding packets). In the following, we survey existing reputation mechanisms, especially focusing on their handling of recommendations.

Some reputation mechanisms do not distinguish between reputation of providing a service and providing a recommendation (e.g., [11]). They assume that trust on an entity's capability to provide services can be transferred to its opinions. For example, in [11], a peer that provides authentic files is trusted to give honest opinions. But such assumption makes the reputation system vulnerable to reputation

[1] `http://www.ebay.com`

[2] `http://www.gnutella.com`

manipulation. For example, a good service provider can exploit it to demote the reputation of its competitors, as its opinions are considered as truthful as its services. Therefore, it is necessary to distinguish the reputation for providing services and recommendations, namely *service reputation* (SRep) and *recommendation reputation* (RRep) respectively. A trustor can evaluate the trustee's *overall reputation* (ORep) based on its *SRep* and others' recommendations. The latter is taken into consideration depending on the recommenders' *RRep*s.

Due to the existence of lies, recommendations need to be carefully incorporated towards the trust decision of whether to interact with a service provider. In another word, a reputation mechanism needs to be able to identify lies such that it is robust against them. Yu and Singh [31] present a reputation model that aims to detect lies in multiple agent systems. Recommendations are compared against the new direct experience to evaluate the recommenders' *RRep*s, which determine the credibility of their recommendations. Only recommendations from helpful nodes (i.e., with high *RRep*s) are accepted and weighed corresponding to their *RRep*s. Similar approach is also taken in [8]. Although they are capable of identifying lies, there is no penalty for either liars or free-riders, which can always benefit from the recommendations of others.

In [30], all recommendations are aggregated to derive the *public opinion*. Each individual recommendation is then compared against the public opinion; too much deviation leads the recommendation to be considered false and thus excluded. The public opinion is then recalculated and compared against each remaining recommendation until no more recommendation is filtered out. This kind of approaches to identify lies are *endogenous* since the truthfulness of recommendations is judged depending on the recommendations themselves. In contrast, *exogenous* approaches use external factors, such as *RRep*, for doing so. The implicit assumption underlying endogenous approaches is that the majority of recommendations are honest such that they dominate the lies. Therefore, a recommendation that deviates from the majority is considered a lie. This assumption is not solid in open environments where recommendations can be very few in number, most of which can be untruthful. A variant of *endogenous* approach is used in [24], where each entity records all the ratings and subsequent interaction experiences. Assume node a receives a recommendation from recommender r, a first picks out all the entities whom r has recommended with a similar value (e.g., within the range [a..b]). The accumulated experiences with those entities are calculated and compared against the rating range to obtain r's *RRep*. Their approach is *exogenous*, because it is the accumulated direct experiences that are used to determine the trustworthiness of a recommendation. Meanwhile, it is also *endogenous* because such comparison is done only within the range of recommendation values that are considered relevant.

Jurca and Faltings [10] propose an incentive-compatible reputation mechanism to deal with inactivity and lies. A client buys a recommendation about a service provider from a special broker named *R-nodes*. After interacting with the provider, the client can sell its feedback to the same R-node, but gets paid only if its report coincides with the next client's report about the same service provider. One issue is that if the recommendation from an R-node is negative such that a client decides to avoid the service provider, the client will not have any feedback to sell. Or in the existence of opportunistic service providers that, for example, behave and misbehave alternatively, an honest feedback does not ensure payback. This opens up the possibility of an honest entity to have negative revenue and thus is unable to buy any recommendation. Besides, the effectiveness of their work depends largely on the integrity of R-nodes, which is assumed to be trusted *a priori*.

In summary, although current reputation mechanisms are capable of identifying lies, they lack measures to enforce voluntary and honest recommending. Therefore, they are not *incentive compatible*, i.e., there does not exist any incentive for entities to actively provide honest recommendations. As there is no deterrent for liars, lies can be rampant and honest recommendations can become difficult to acquire due to lack of motivation. Therefore, a distributed reputation mechanism for ubiquitous computing environments not only needs to be robust against lies, but also needs to enforce both active and honest recommendation. In the following, we present such a reputation mechanism, starting with reputation representation.

3. REPUTATION REPRESENTATION

Since reputation essentially aggregates past experiences and dynamically evolves, it bears great similarity with Bayesian analysis, which is a statistical procedure that estimates parameters of an underlying distribution based on observations. An extensively used distribution in Bayesian analysis is Beta distribution.

3.1 Beta Distribution

According to the probability theory, the posterior probability for binary events can be estimated by beta distribution. For example, given a process with two possible outcomes (T, $\neg T$), let r, s be the observed number of T and $\neg T$ respectively, the *Probability Density Function* (PDF) of the probability p of having the outcome T for the next time can be given by beta distribution (with $\alpha = r+1$ and $\beta = s+1$):

$$\begin{aligned} f(p|\alpha,\beta) &= \frac{1}{B(\alpha,\beta)} p^{\alpha-1}(1-p)^{\beta-1}, \\ &\quad \text{where } 0 \le p \le 1, \alpha, \beta \ge 0 \\ B(\alpha,\beta) &= \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt \end{aligned}$$

where α and β are two parameters used to index the continuous family of Beta distribution and $B(\alpha,\beta)$ is the beta function. $f(p|\alpha,\beta)$ represents a probability distribution of p in terms of integrals. Formally, the probability of p falling into $[a,b]$ is $\int_a^b f(p|\alpha,\beta)dp$. The prior distribution (the initial state) is $f(p|1,1)$, leading to uniform distribution (Figure 1). It reflects the fact that without any knowledge, the probability of having T for the next time can be any value between 0 and 1 with equal possibility. New observations are used to update the PDF of p. For example, having observed 8 times T and 2 times $\neg T$, the PDF can be expressed as $f(p|9,3)$, as plotted in Figure 1.

The expected (mean) value of the beta distribution $f(p|\alpha,\beta)$ assumes a simple form:

$$\mathrm{E}(p) = \frac{\alpha}{\alpha+\beta}$$

It gives the mean value of p, based on $(\alpha+\beta-2)$ observations accumulated so far. For example, in Figure 1, the expected values of both $f(p|9,3)$ and $f(p|21,7)$ equal to 0.75. It can be interpreted as that the probability of observing outcome T in the future is uncertain, but the expected value is 0.75. In addition, $f(p|21,7)$ has more confidence saying so (i.e., $f(0.75|21,7) > f(0.75|9,3)$), thanks to more accumulated observations.

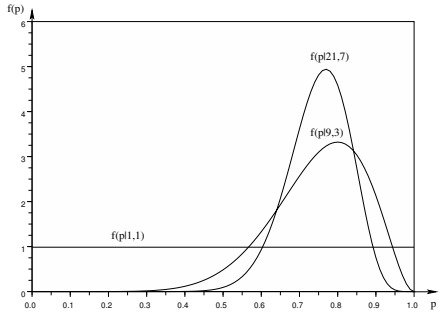

Figure 1: Beta Distribution values

3.2 Beta Reputation

As reputation is essentially an *a posteriori* estimation based on historic experiences (either direct or indirect), beta distribution has been recognized as a useful model to model reputation [21, 9, 3]. Therefore, we represent reputation based on beta distribution (abbreviated as *beta reputation*). A reputation value assumes a tuple of (α, β) $(\alpha,\ \beta \geq 1)$, with α and β representing positive and negative experiences respectively.

As beta distribution only considers binary events, it is not enough to describe the experience of service consumption, which can fall into the range between being completely satisfactory and completely unsatisfactory. Therefore, an experience is evaluated with *Quality of Experience* (QoE), saying, between 0 (completely unsatisfactory) and 1 (completely satisfactory). This experience is split into two parts: QoE contributing to the positive experience and $(1-QoE)$ to the negative experience. Therefore, beta reputation $f(p|\alpha,\beta)$ gives the PDF of *the probability of having a complete satisfactory experience*, i.e., the expected QoE.

Thanks to sound statistical properties of beta distribution, beta reputation has the following advantages:

1. It is easy to assess the trustworthiness of an entity with reputation of (α,β), i.e., by calculating $\frac{\alpha}{\alpha+\beta}$.

2. It is easy to evaluate how many experiences (i.e., $\alpha+\beta-2$) have contributed to the current reputation. The larger this value is, the more probably the reputation assumes the expected value. Only newcomers' reputation is based on 0 experience.

3. It facilitates the combination of experiences from multiple sources, including the trustor itself and different recommenders. This is because the add operation of beta reputation is straightforward: $f(p|\alpha_1,\beta_1) + f(p|\alpha_2,\beta_2) = f(p|\alpha_1+\alpha_2, \beta_1+\beta_2)$.

4. It reflects the nature of reputation, which is the aggregation of observations. An entity dynamically adjusts the reputation with more experiences being accumulated, which is similar to deriving posterior distribution after observations are made.

5. It captures the uncertainty of reputation. Beta distribution only gives the PDF of the probability of having an outcome, which matches the fact that reputation can only give probabilistic estimation of an entity's future behavior.

Alternatively, reputation can be also represented with a single value from discrete (e.g., [1]) or continuous value space (e.g., [18]). Compared to beta reputation, single-value based reputation representation does not reflect the amount of experiences that contribute to the reputation. In addition, with single value based reputation, *ignorance*, which refers to the reputation without any knowledge, generally bears the value of 0. It can not be distinguished from the 0 reputation values that result from a mixture of positive and negative experiences (e.g, [18, 21]). While with beta reputation, only newcomers have a reputation of $(1,1)$.

Beta distribution's feature of easy experience aggregation facilitates the derivation of an entity's reputation, which is formed based on the trustor's direct experiences and others' recommendations, as explained as follows.

4. REPUTATION FORMATION

Before we proceed to show how reputation is formed, we first explain the notations to be used in the reputation mechanism. As reputation is always about an entity o (i.e., trustee) held by some entity a (i.e., trustor), we denote o's reputation from the point of view of a as $Rep_a(o)$. Table 1 lists the notations we use, including service reputation ($SRep$), recommendation (Rec), recommendation reputation ($RRep$) and overall reputation ($ORep$). They are expressed using beta reputation, with two parameters representing positive and negative experiences respectively.

Table 1: Notations in the reputation mechanism

Label	Value Range	Meaning
$SRep_a(o)$	(s_p, s_n)	a's direct experiences with o
$Rec_a(o)$	(c_p, c_n)	Recommendation made by node a regarding node o. Helpful recommenders give recommendations based on their own direct experiences, i.e., $Rec_a(o) = SRep_a(o)$
$RRep_a(o)$	(r_p, r_n)	Recommendation reputation of node o held by node a
$ORep_a(o)$	(o_p, o_n)	Overall reputation of node o held by node a

Each node keeps both $SRep$ and $RRep$ of its *acquaintances*, the entities with which it has interacted before (either as a service client or a recommendation requester), as shown in Table 2. This table of acquaintance records is named *acquaintance table*. In the table, aID denotes

Table 2: An entry of the acquaintance table

aID	SRep			RRep		
	s_p	s_n	t_s	r_p	r_n	t_r

acquaintance ID and t_s and t_r represent respectively the timestamps when the $SRep(s_p, s_n)$ and $RRep(r_p, r_n)$ were updated last time.

ORep can rely solely on the trustor's direct experiences (i.e., *SRep*) if they are significant enough to derive a trust decision. This can be judged by checking whether the total accumulated $(s_p + s_n - 2)$ experiences reach a certain threshold. Otherwise, it asks for recommendations from others. The recommendations and the node's own direct experiences are then combined to evaluate the overall reputation (*ORep*) of the trustee. Assuming a recommender r gives a recommendation regarding o (i.e., $Rec_r(o)$) to client c and $RRep_r(c) = (r_p, r_n)$, the recommendation is considered trustworthy and accepted if (1) r is honest enough, by checking whether $\frac{r_p}{r_p+r_n}$ is high enough and (2) the *RRep* is evaluated based on enough evidences by checking whether (r_p+r_n-2) is large enough. If the recommendation is taken into account, it is given a weight $w_r = \mathrm{E}(Beta(r_p, r_n)) = \frac{r_p}{r_p+r_n}$.

The weights of different recommendations are further normalized by dividing with the sum of all weights. Therefore, *ORep* can be evaluated using *SRep* and the recommendations from helpful recommenders:

$$ORep = \delta \times SRep + (1-\delta) \times \frac{\sum_{r \in R}(Rec_r(o) \times w_r)}{\sum_{r \in R}(w_r)} \quad (1)$$

where δ is the weight given to its direct experience (*SRep*) and is generally greater than 0.5. The favor of direct experiences over recommendations is due to the fact that entities tend to rely on their own experiences more than on others' recommendations [13]. Therefore, an entity can make a trust decision based on the overall reputation (*ORep*) of the trustee. *ORep* is not kept as a field of the acquaintance record, instead it is dynamically evaluated when needed, since it evolves with time and new experiences.

5. REPUTATION EVOLUTION

An entity can change its behavior over time, making old experiences become irrelevant for the actual reputation evaluation [9, 15]. This calls for discount of past, which gives more weight to recent experiences than old ones. Such discounting also prevents an entity from capitalizing on its previous good behavior forever. Hence, reputation fades with time, as shown as follows.

5.1 Time Fading

Since both recent behaviors and past histories contribute to the reputation, their assigned weights decide how fast the reputation builds up. For example, if recent behavior is assigned a very high weight, a node's reputation tears down very fast after a few misbehaviors. We assign more weight to recent behavior, as suggested by the studies in [5].

Given the time interval of Δt, the reputation (α, β) evolves after every Δt:

$$\begin{aligned} \alpha' &= 1 + (\alpha - 1) \times \rho^{\Delta T} \\ \beta' &= 1 + (\beta - 1) \times \rho^{\Delta T} \end{aligned}$$

where ρ is *time fading factor*, whose value falls into the range of [0..1]. The lower value ρ has, the more quickly histories are forgotten. When ρ equals 0, histories are immediately forgotten; while when ρ equals 1, the history is forever kept and considered equivalent regardless of age.

As shown in the above equations, when $\Delta T \rightarrow +\infty$, $\rho^{\Delta T} \rightarrow 0$. It expresses the fact that inactivity between two parties for a long time leads to complete discount of experience, making its reputation the same as that of newcomer. This is because the experiences can be too old to be indicative of the trustee's trustworthiness. Both *SRep* and *RRep* fade according to the above equations. For simplicity, the reputation value in the rest of this paper does not bear a timestamp and it always refers to the current reputation unless indicated otherwise.

Reputation also evolves with new experiences, as reputation aggregates the overall experiences with an entity. This is reflected in both *SRep* and *RRep*, which aggregate the experiences of consuming services and utilizing recommendations respectively.

5.2 Evolution of Service Reputation (SRep)

Since $SRep(s_p, s_n)$ combines all direct experiences, it is updated whenever a new experience occurs. An experience is described with the metric of Quality of Experience (*QoE*). As the goal of the reputation mechanism is to identify dishonest service providers that do not comply with their advertised QoS, *QoE* is accordingly measured based on the QoS conformance of the service provider. More specifically, given n QoS dimensions of d_i $(i = 1..n)$ (e.g., *availability*, *latency*) which client a cares about, service provider o states in its service advertisement $(p_1, p_2, .., p_n)$ in which p_i is the promised value for dimension d_i. After the service completes, the QoS that a receives is represented by $(a_1, a_2, .., a_n)$, in which a_i is the actual value for dimension d_i. The $QoE_a(o)$ can be assessed by:

$$QoE = \sum_{1 \le i \le n} comp(a_i, p_i)/n \quad (2)$$

where $comp(a_i, p_i)$ is a function to calculate one-dimension degree of conformance between the actual and promised QoS. Depending on the QoS dimension, it assumes the following forms:

1. $comp(a_i, p_i) = MIN(1, a_i/p_i)$ when dimension i is quantitative and stronger with larger values, for example, *availability*.
2. $comp(a_i, p_i) = MIN(1, p_i/a_i)$, when dimension i is quantitative and stronger with smaller values, for example, *latency*.
3. $comp(a_i, p_i) = 1 - (a_i \otimes p_i)$ when dimension i is qualitative and bears Boolean values, for example, confidentiality. $\otimes$ represents XOR function, i.e., $x \otimes y = 0$ if x equals y, and 1 otherwise.
4. For dimensions whose value space is literals (e.g., *service adaptation policy*), $comp(a_i, p_i)$ equals 1 when the required policy is satisfied, 0 otherwise.

For example, given a service provider's advertisement of ($latency$ = 0.8 ms, $availability$ = 99%), a service client's actual experienced QoS is ($latency$ = 1.0 ms, $availability$ = 100%), then $QoE = (MIN(1, 0.8/1.0)+MIN(1, 100\%/99\%))/2 = 0.9$.

With a new QoE, the $SRep(s_p, s_n)$ is updated as described in Section 3.2: (i) $s'_p = s_p + QoE$; (ii) $s'_n = s_n + (1 - QoE)$.

5.3 Evolution of Recommendation Reputation (RRep)

Similarly, *RRep* dynamically evolves with recommendations being elicited and services being consumed. A recommendation bears the form of (c_p, c_n), which is equal to *SRep* for an honest recommender. Given a new QoE of $e \in [0..1]$, the honesty of a recommender is adjusted according to the helpfulness of its recommendation.

$$\Delta e = \int_{MAX(0,e-0.4)}^{MIN(e+0.4,1)} f(p|c_p, c_n)dp$$

At first, Δe evaluates the probability of having a QoE in the range of $[MAX(0, e - 0.4), MIN(e + 0.4, 1)]$, according to the recommendation of (c_p, c_n). For example, in Figure 2, if a new experience e equals 0.8 and the recommendation is $(4, 2)$, Δe is equal to the size of the shaded area. It is compared against the probability if the trustor has no knowledge about the trustee, i.e.,

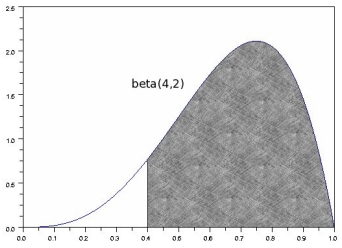

Figure 2: Calculation of Δe

$$\Delta min = \int_{MAX(0,e-0.4)}^{MIN(e+0.4,1)} f(p|1, 1)dp \qquad (3)$$

Therefore, a recommendation with Δe larger than Δmin is considered helpful, and unhelpful otherwise. The helpfulness of the recommendation can be evaluated with

$$e' = MAX(MIN(\Delta e - \Delta min + 0.5, 1.0), 0.0)$$

where the MAX and MIN operators are used to ensure that e' falls into [0..1]. The $RRep(r_p, r_n)$ is then updated accordingly: (i) $r'_p = r_p + e'$; (ii) $r'_n = r_n + (1 - e')$.

Assume that before a client c has a new experience (e) of 0.8 with service provider o, it has received recommendations of $(2, 4)$ and $(4, 2)$ from two recommenders a and b respectively. As $\Delta min = 0.6$ (using Equation 3), the helpfulness of a's recommendation is $e' = 0.24$ and b's recommendation leads to $e' = 0.81$. It thus enables distinguishing between honest and dishonest recommenders, as well as different degree of honesty/dishonesty.

Based on the features of beta reputation, the value of $(r_p + r_n - 2)$ is high if an entity is active in providing recommendations; the expected value of $f(p|r_p, r_n)$ is high if an entity is honest in doing so. With two values δ_h and δ_a defined as threshold trustworthiness and activeness in providing recommendations, a recommender with $RRep(r_p, r_n)$ is considered active if $r_p + r_n - 2 \geq \delta_a$, and inactive otherwise; it is considered honest if $\frac{r_p}{r_p+r_n} \geq \delta_h$, and dishonest otherwise. It leads to 5 possible states of a recommender: active truth-teller (AT), inactive truth-teller (IT), active liar (AL), inactive liar (IL) and newcomer (Figure 3).

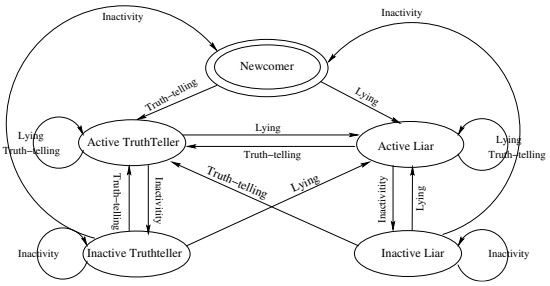

Figure 3: The states of a recommender

A recommender can convert from one state to another, depending on its behavior. An active truth-teller enforces its state by continuing recommending honestly and weakens its state by lying. If it continues lying, with the fading of previous good behavior, the accumulated experiences will eventually work against it and degrade it to an *active liar*. If a recommender has not provided any recommendation for so long a time that its *RRep* decays, it is considered as an inactive recommender and even a *newcomer*.

Note that an active and honest recommender can be considered inactive due to the fact that it does not have any direct experience with the trustee being evaluated by the recommendation requester. Therefore, although inactivity can result from an entity's withholding recommendations on purpose, it does not necessarily infer free riding. But in order to motivate a node to become an active truth-teller, active and honest recommenders should be able to have better success in identifying dishonest entities using the reputation mechanism. This is realized during reputation propagation, where different recommenders are treated differently in terms of their accessibility to helpful recommendations.

6. REPUTATION PROPAGATION

Lack of enough direct experiences triggers a trustor's elicitation of recommendations (e.g., by broadcasting the request) from nearby entities. Of all the collected recommendations, only those from truth-tellers (i.e., honest recommenders) are taken into account. If there is no recommendation from any truth-teller, the trustor takes into consideration those from inactive and first-time encountered recommenders by calculating their average. With the recommendations from others, the trustor evaluates the trustee's *ORep* using Equation 1. Otherwise, the trustor has to rely on its direct experiences which are too few to make a sound decision. The trust decision will then have to be made depending on other factors, e.g., the trustor's attitude towards strangers. If the trust decision leads to a service consump-

tion and thus a new QoE, the latter is compared against all recommendations to update the recommenders' $RRep$s.

A trustor elicits recommendations indiscriminately but accepts only those from honest recommenders. This is for the purpose of ensuring robustness against lies, while empowering the trustor with the capability of recognizing new recommenders and continuously updating a dishonest recommender's $RRep$. More specifically, even though the recommendations from liars are not taken into account, they are used to update the $RRep$s of the recommenders, which can be improving if they become honest or deteriorating if they continue lying.

Whenever an honest recommender a receives a request for recommendation regarding an entity o, it first checks whether its direct experiences with o are significant enough for recommending. If that is not the case, a does nothing as it cannot be of any help. Otherwise, i.e., a has enough direct experiences for recommending, it handles the request depending on the state of the recommendation requester:

- If the requester is considered as an active truth-teller, a sends back its $SRep_o(a)$ immediately.
- If the requester is considered as an active liar, a simply ignores the request.
- If the requester is considered inactive, a gives back its recommendation with a probability depending on $diff = \delta_a - (r_p + r_n - 2)$. Inactive recommenders are better treated than liars due to the fact that inactive recommenders do not necessarily withhold their recommendations. To distinguish inactive truth-tellers (IT), newcomers and inactive liars (IL), the IT and IL's probabilities are increased and decreased with an small value of ϵ respectively. Therefore, the less active an entity is, the less possible that it receives helpful recommendations from others. Note that newcomers also suffer from low probability of eliciting honest recommendations.

The reason of honest recommenders' going through the above process of treating different type of recommendation requesters differently is that if honest recommenders are over-generous by treating everybody alike, other entities are not motivated to return the favor because doing that does not give them any advantage. Eventually, rational entities choose to withhold their recommendations while liars remain unpunished. Honest recommenders will suffer by having less and less useful recommendations from others and will eventually draw no favor back. Therefore, honest recommenders have to assume the defensive strategy of holding 'grudges' against liars by keeping others' $RRep$ in order to guard their own interests. In contrast, dishonest recommenders, i.e., liars, treat all types of recommendation requesters alike since their utmost goal is to spread lies as widely as possible to take advantages such as promoting their colluders' reputation.

The deterrent for nodes to lie thus lies in the consequence of lacking helpful recommendations from honest recommenders, who hold grudges. At the same time, refusing to provide recommendations leads to less accessibility to helpful recommendations. Lack of recommendations forces a trustor's trust decision to be solely dependent on its direct experiences, which often can be too few or old to be helpful in open environments. This causes wrong trust decisions, making the client either interact with dishonest service providers or avoid honest ones. It is more clearly demonstrated in the next section, which evaluates the performance of our reputation mechanism.

7. REPUTATION MECHANISM EVALUATION

In this section, we evaluate the performance of our reputation mechanism in helping nodes distinguish honest and dishonest service providers, and to identify honest and active recommenders, based on simulations.

7.1 Experiment Setting

The simulation is carried out with Network Simulator (ns-2) with CMU wireless extensions [14]. Our simulated network consists of 40 mobile nodes in an area of 400m × 400m, with each node having a transmission range of 100m. We use the *Random Waypoint* mobility model with each node moving at walking speed, i.e., between 0.5 m/s and 1.5 m/s, with pause time of 0, i.e., nodes are always in motion. The Distributed Coordination Function (DCF) of the IEEE 802.11 protocol is used as the MAC layer protocol. The underlying routing protocol is Optimized Link State Routing Protocol (OLSR) [4]. The simulation parameters are summarized in Table 3.

Table 3: NS-2 simulation parameters

Parameter	Value
Mobility Model	Random Way Point
Moving Speed	0.5 - 1.5 m/s
Pause Time	0
Transmission Range	100 m
Area	400m x 400m
Number of Nodes	40
Routing Protocol	OLSR

The population of 40 nodes includes 8 types of entities with different behavior in service providing (honest or not), recommendation providing (honesty and activeness), as shown in Table 4. Each type of entity has the same population, i.e., 5 each (different settings with different population sizes will also be studied later).

Table 4: The types of nodes with different behavior

Type	Service Providing Honesty	Recommendation	
		Honest	Active
1	+	+	+
2	+	+	−
3	+	−	+
4	+	−	−
5	−	+	+
6	−	+	−
7	−	−	+
8	−	−	−

For simplicity without losing generality, we assume that every node can be the service provider for the other. Start-

ing from time 50[3], for every 1 second, a node makes a trust decision whether to interact with a random node in its routing table, in a round robin way. A trust decision is made as follows: (1) the service client first checks whether the *SRep* of the service provider is based on enough experiences (threshold $\delta_a = 1.0$); (2) if yes, it calculates whether the expected value of *SRep* reaches a threshold value ($\delta_h = 0.6$); (3) if not, it elicits recommendations from its neighbors (we set the request broadcast range to 2 hops). If the aggregation of *SRep* and others' recommendations are still not enough for making a trust decision, the node decides whether to interact with a service provider with a certain probability. In our experiments, the probability is set to 1.0, assuming an optimistic attitude facing uncertainty.

A total of 60 rounds have been executed. An honest service provider offers a QoE of 0.9, while a dishonest one offers a QoE of 0.1. Honest recommenders recommend with their $SRep(s_p, s_n)$ regarding the trustee, while dishonest recommenders send back lies which are complementary to their *SRep*s, i.e., a recommendation assumes the value of $(r_p = s_n, r_n = s_p)$. Active recommenders offer recommendations with 90% probability, while inactive ones offer with 10% probability.

We investigate and compare the performance of the 4 different types of recommenders: active truth-teller (type 1 + type 5 in Table 4), inactive truth-teller (type 2 + type 6), active liar (type 4 + type 8) and inactive liar (type 3 + type 7). The advantage of being an active truth-teller is reflected in the fact that they can elicit more honest recommendations, which help them make right trust decisions regarding whether to interact with an entity or not. Therefore, we show (1) the number of honest recommendations obtained by the four types of recommenders respectively. When a client fails to acquire any helpful recommendation, it has to base its trust decision solely on its direct experiences, which are not significant enough for a sound decision. Namely, the client is subject to *mistakes*, i.e., wrong trust decisions, which refer to either false positives (when an honest service provider is identified as an untrustworthy one) or false negatives (when a dishonest service provider is not identified as being so). Thus, (2) the number of mistakes made by different recommenders are also displayed. These metrics are recorded every 200 seconds to show the evolution of reputation. They are detailed below.

7.2 Evaluation Results

7.2.1 Elicited Honest Recommendations

Figure 4 shows the number of elicited honest recommendations for different type of recommenders. It can be observed that at the beginning (before time 500s), very few recommendations are propagated and the four types of recommenders do not have much difference in the number of obtained honest recommendations. With the accumulation of experiences, the honest entities have enough experiences to recommend. Recommendation reputation is gradually recognized and the order of benefit (AT > IT > IL > AL) starts to be established, from time 1500s in Figure 4.

7.2.2 Mistakes

[3]This aims to give OLSR enough time to build routing table, as OLSR is a proactive protocol.

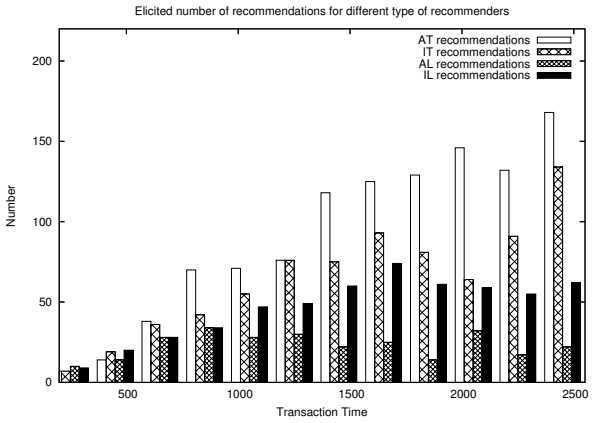

Figure 4: Number of elicited honest recommendations

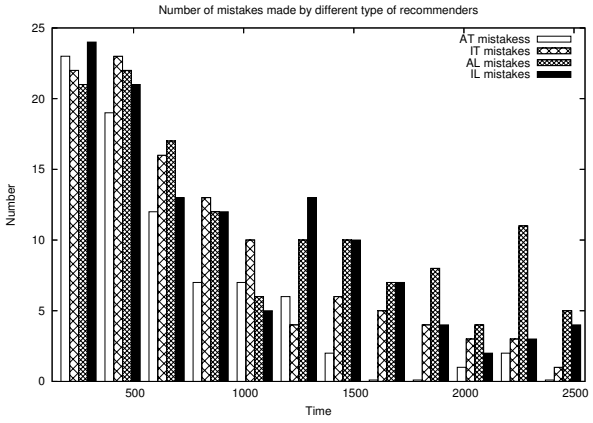

Figure 5: Number of made mistakes

Lack of honest recommendations can lead to mistakes. Figure 5 presents the number of mistakes made by the four types of recommenders. It can be seen that, at the beginning, every type of nodes make mistakes as many as half of the total transaction number. It is because most decisions are blind and honest service providers occupy half of the population.

With more accumulated experiences, every type of recommenders make less and less mistakes. Especially, with the help of honest recommendations, AT nodes make the least number of mistakes and AL nodes make the most (the order of AT > IT > IL > AL is enforced). Note that dishonest or inactive recommenders can also tell the honesty and activeness of a recommender using reputation system. However, they have access to less number of truthful recommendations for making the right trust decision.

In order to demonstrate more clearly the advantages brought by useful recommendation, the percentages of mistakes out of all transactions for different recommenders are shown in Figure 6. It can be seen that, starting from time 1500s, ATs make less than 5% of mistakes while ALs suffer more than 20% of mistakes.

7.2.3 Other Results

In the above simulations, we have set the population size

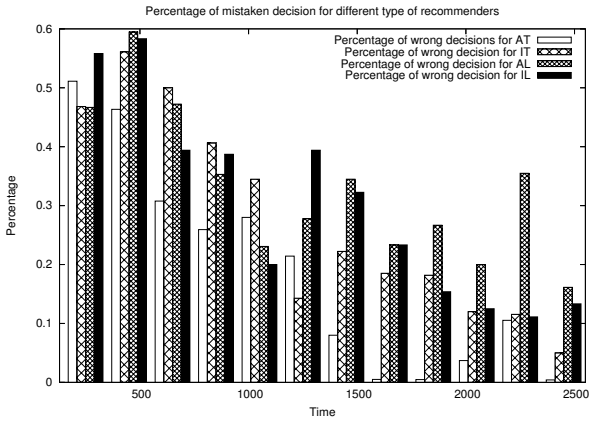

Figure 6: Percentage of wrong trust decisions

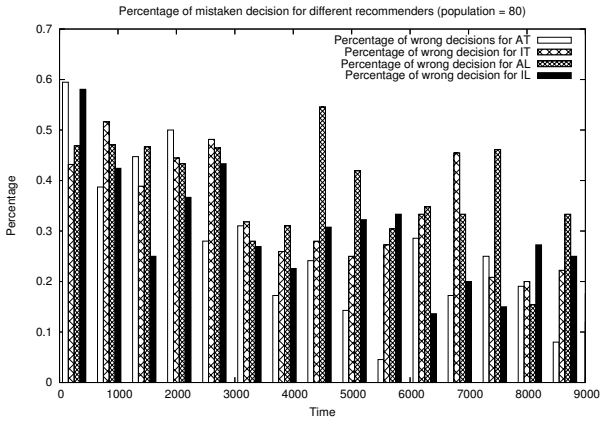

Figure 7: Percentage of wrong trust decisions with larger population

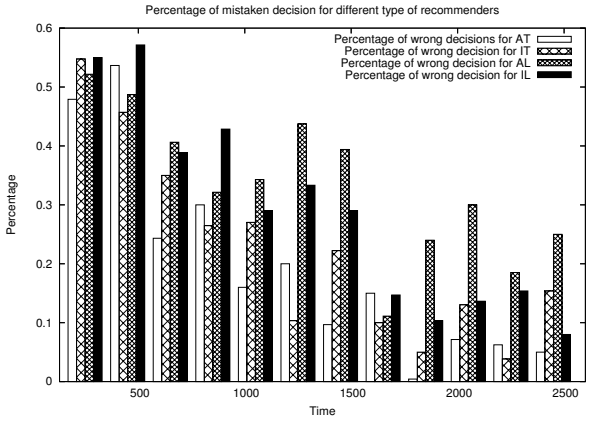

Figure 8: Percentage of wrong trust decisions with different population composition

to be relatively small (40) to lessen the time for bootstrapping (about 1500 seconds for 40 nodes), because nodes need to acquire experiences to be able to give useful recommendations. Basically, a larger population takes longer time to bootstrap, but the reputation mechanism shows similar effects. We did similar simulations with larger population (80) and the percentage of mistakes is shown in Figure 7. It can be observed that the order of benefit is established eventually, although it takes more time than in a community of 40 nodes (about 3500 seconds into simulation).

The reputation mechanism also exhibits similar performance with different population percentage of different recommenders. We have carried out the simulations by decreasing the population of active truth-teller (AT) to 10% and increasing active liar (AL) to 40%. The percentages of mistakes for different recommenders are presented in Figure 8. It shows that the order of the treatment (i.e., AT $>$ IT $>$ IL $>$ AL) is also established.

8. CONCLUDING REMARKS

In this paper, we have presented a distributed reputation mechanism for recognizing the trustworthiness of a service provider in ubiquitous computing environments, including reputation representation, formation, evolution and propagation. Our contribution includes: (1) proposing a simple yet effective reputation mechanism that not only is lie-proof, but also motivates active and truthful recommendation sharing; (2) modeling a reputation that continuous evolves, with time and with new experiences; (3) evaluating the effectiveness and performance of the proposed reputation mechanism via simulation tests.

As an entity has to handle the reputation independently and autonomously, in our reputation mechanism, it stores the reputation values of all of its acquaintances. This might raise an issue if the population of the acquaintances is so large such that it brings considerable overhead for reputation storage and manipulation. A possible solution is to manage nodes by groups, each of which share a group reputation [26, 21]. The reputation of an entity depends on the group it belongs to; the behavior of a member affects the reputation of its group. This requires strong group support [16], as the group members need to trust each other and have common interests such that they are motivated to protect the group's reputation.

An important issue in reputation mechanism is identity changing. Most online reputation systems protect privacy and each agent's identity is normally a pseudonym. It causes problems because pseudonym can be changed easily [21]. When a user ends up having a reputation lower than that of a new comer, she can discard her initial identity and start from the beginning. This calls for the necessity of special treatments of newcomers. We partly address this issue by putting newcomers in a unfavorable position, such that they have difficulties obtaining helpful recommendations, until they accumulate enough good behavior. But to completely solve this issue, it would have to rely on other mechanisms, such as introducing an "entry fee" for each pseudonym or use of once in a lifetime pseudonym that is bound to a real-world entity [7]. A very related issue is called Sybil attack [6]: if there is no control over creation of new entities, a real-world entity can create as many identities as it wishes to challenge the use of majority in reputation systems. The only challenge this attack can bring to our reputation system is when there is no recommendation from an active truth-teller, the trustor relies on the average of all recommendations from unknown (or barely known) recommenders. The addressing of these challenges will be part of future work.

9. ACKNOWLEDGEMENTS

This research is partially supported by the European IST PLASTIC project[4] (EU-IST-026955).

10. REFERENCES

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proc. Hawaii Int'l Conf. System Science HICSS-33*, 2000.

[2] C. Adams and S. Farrell. *RFC2510 - Internet X.509 Public Key Infrastructure Certificate Management Protocols*, 1999.

[3] S. Buchegger and J.-Y. L. Boudec. A robust reputation system for P2P and mobile ad-hoc networks. In *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[4] T. Clausen and P. Jacquet. Optimized link state routing protocol. IETF RFC 3626, 2003.

[5] C. Dellarocas. The digitization of word-of-mouth: promise and challenges of online feedback mechanisms. MIT Working Paper, March 2003.

[6] J. Douceur. The sybil attack. In *Proc. of the IPTPS02 Workshop*, 2002.

[7] E. Friendman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2), 2001.

[8] T. D. Huynh, N. R. Jennings, and N. Shadbolt. On handling inaccurate witness reports. In *Proc. 8th International Workshop on Trust in Agent Societies*, pages 63–77, Utrecht, The Netherlands, 2005.

[9] A. Josang and R. Ismail. The beta reputation system. In *Proc. of 15th Bled Conf. on Electronic Commerce*, 2002.

[10] R. Jurca and B. Faltings. An incentive compatible reputation mechanism. In *Proc. of IEEE International Conference on E-Commerce*, 2003.

[11] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International World Wide Web Conference*, 2003.

[12] T. Kindberg and A. Fox. System software for ubiquitous computing. *IEEE Pervasive Computing*, 1(1):70–81, 2002.

[13] P. Kollock. The emergence of exchange structures: an experimented study of uncertainty, commitment, and trust. *American Journal of sociology*, 100(2), 1994.

[14] LBNL. Network simulator ns-2. `http://www.isi.edu/nsnam/ns`, 2001.

[15] J. Liu and V. Issarny. Enhanced reputation mechanism for mobile ad hoc networks. In *Proc. of Int'l Conf. on Trust Management (iTrust)*, 2004.

[16] J. Liu, F. Sailhan, D. Sacchetti, and V. Issarny. Group management for mobile ad hoc networks: Design, implementation and experiment. In *Proc. of the 6th Int'l Conf. on Mobile Data Management*, 2005.

[17] IETF working group: Mobile ad hoc networks (manet). `http://www.ietf.org/html.charters/manet-charter.html`, 2005.

[18] S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994.

[19] P. Michiardi and R. Molva. CORE: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *CMS'2002*, 2002.

[20] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting honest feedback in electronic markets. Working Paper, August 2002.

[21] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *Proceedings of the 35th HICSS*, 2002.

[22] OASIS. Service oriented architecture reference model. Working Draft, `http://xml.coverpages.org/SOA-RM-WD05.pdf`, 2005.

[23] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Communications of the ACM*, 46(10), 2003.

[24] J. Patel, W. L. Teacy, N. R. Jennings, and M. Luck. A probabilistic trust model for handling inaccurate reputation sources. In *Proc. of 3rd Int'l Conf. on Trust Management (iTrust 2005)*, 2005.

[25] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[26] J. Sabater and C. Sierra. Regret: A reputation model for gregarious societies. In *Proc. 4th Workshop Deception, Fraud, and Trust in Agent Societies*, 2001.

[27] M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 2001.

[28] R. Sen, R. Handorean, G.-C. Roman, and C. Gill. *Service-Oriented Software System Engineering: Challenges and Practices*, chapter Service Oriented Computing Imperatives in Ad Hoc Wireless Settings, pages 247 – 269. Idea Group, 2005.

[29] M. Weiser. The computer for the 21st century. *Scientific American*, September 1991.

[30] A. Whitby, A. Josang, and J. Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proceedings of the 7th Int'l Workshop on Trust in Agent Societies*, 2004.

[31] B. Yu and M. P. Singh. Detecting deception in reputation management. In *Proceedings of the Second International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, pages 73–80, 2003.

[4] `http://www.ist-plastic.org`